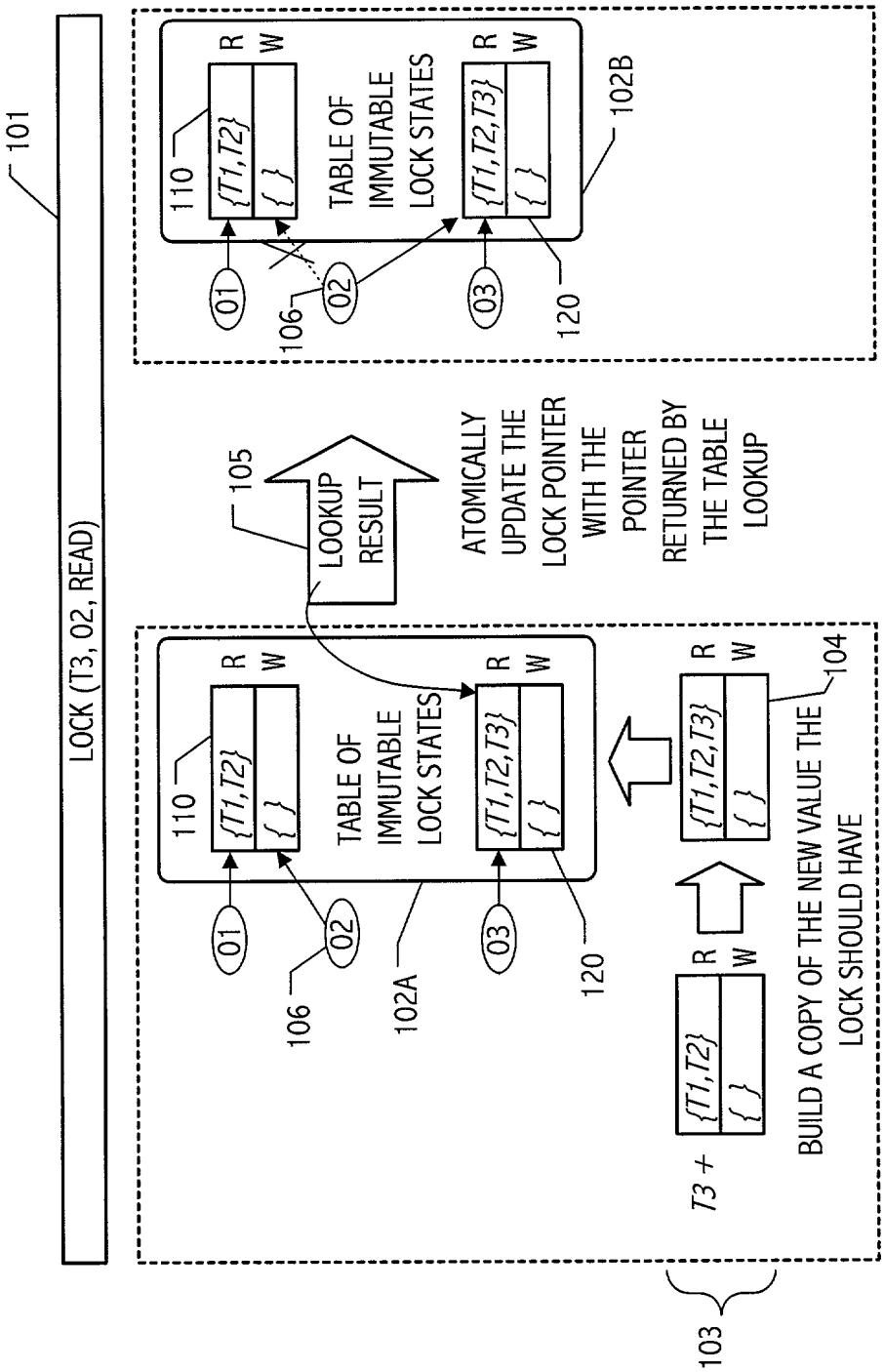
101
LOCK (T3, 02, READ)
110
{T1,T2}
{ }
R
W
TABLE OF IMMUTABLE LOCK STATES
{T1,T2,T3}
{ }
102B
01
02
03
106
120
105
LOOKUP RESULT
ATOMICALLY UPDATE THE LOCK POINTER WITH THE POINTER RETURNED BY THE TABLE LOOKUP
102A
T3 +
104
BUILD A COPY OF THE NEW VALUE THE LOCK SHOULD HAVE
103

FIG. 1

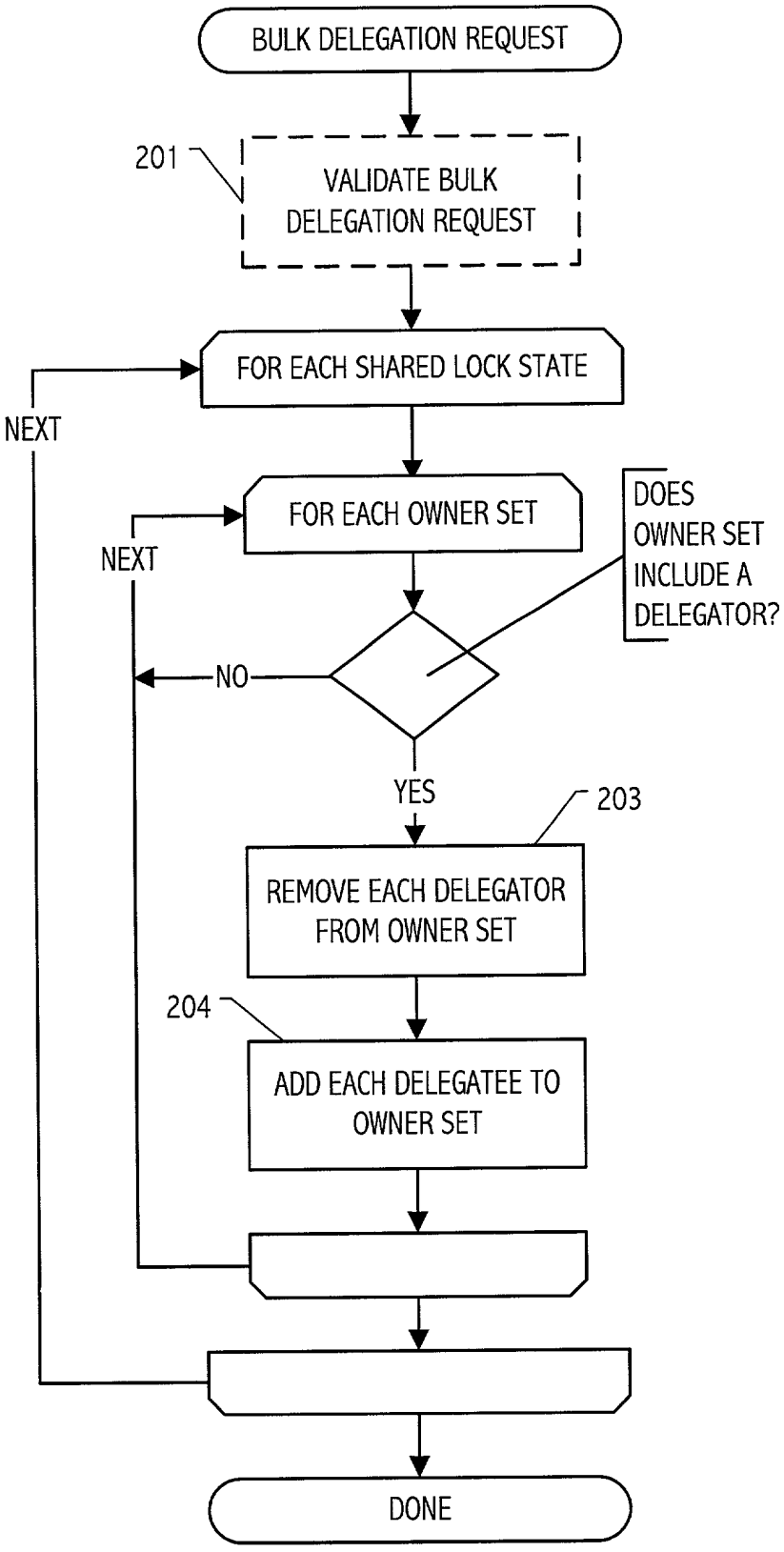
BULK DELEGATION REQUEST
201
VALIDATE BULK DELEGATION REQUEST
FOR EACH SHARED LOCK STATE
NEXT
FOR EACH OWNER SET
NEXT
DOES OWNER SET INCLUDE A DELEGATOR?
NO
YES
203
REMOVE EACH DELEGATOR FROM OWNER SET
204
ADD EACH DELEGATEE TO OWNER SET
DONE

FIG. 2

```
delegate(delegators, delegatees)
begin
  foreach l in TSLS
    if ∃ M, (delegators ∩ Owners(l, M) ≠ ∅) ∧ (∄ M_i, M_i > M ∧ (delegators ∩ Owners(l, M_i) ≠ ∅))
      TSLS.remove(l)
      // modify its owner set to reflect the effect of delegation
      foreach M_i, M_i ≤ M
301 ~~~~  Owners(l, M_i) ← [Owners(l, M_i) − delegators] ∪ delegatees
      end
      // does the new value duplicate an existing shared lock state ?
      if TSLS.contains(l)
        // yes. record the "original" being duplicated
        // and add the shared lock state to the set of duplicates.
        original(l) ← TSLS.get(l)
        duplicates.add(l)
      else
        // no. Re-enter the modified shared lock in the TSLS.
        TSLS.add(l)
      endif
    endif
  end
  // Process duplicates now.
  foreach l in duplicates
    if ∃ M, (delegators ∩ Owners(l, M) ≠ ∅) ∧ (∄ M_i, M_i > M ∧ (delegators ∩ Owners(l, M_i) ≠ ∅))
      // modify its owner set to reflect the effect of delegation
      foreach M_i, M_i ≤ M
        Owners(l, M_i) ← [Owners(l, M_i) − delegators] ∪ delegatees
      end
    endif
  end
end
```

FIG. 3

Atty. Dkt. No. 004-5639
1st Inventor: Laurent P. Daynes
Attorney: David W. O'Brien, (512) 347-9030

```
// Determine the validity of a delegating a lock set to the value l
boolean isValid(delegators, delegatees, l)
begin
  if Owners(l, Write) = ∅
    return true
  endif
  // at least one write lock owner
  if Owners(l, Write) ∩ delegators = ∅
    // All the delegators are read owners.
    // The delegation is valid if all delegatees can ignore read-write
    // conflicts with the write owners.
    return ∀t ∈ delegatees, Owners(l, Write) ⊆ ICW(t, rw)
  endif
  // the lock is delegated in write mode – all delegatees must ignore
  // write-write conflicts between each others and with each remaining
  // owners of the lock in write mode. Also, write-read conflicts should
  // be ignored with remaining owners of the lock in read mode.
  if | delegatees |> 1
    // More than one delegatee
    if ∃ t ∈ delegatees, ∃ c ∈ {rw, wr, ww}, delegatees ⊈ ICW(t, c)
      return false
    endif
  endif
  if ∃ t ∈ delegatees, (Owners(l, Write) – delegators) ⊈ ICW(t, ww)
    return false
  endif
  if ∃ t ∈ delegatees, (Owners(l, Read) – delegators) ⊈ ICW(t, wr)
    return false
  endif
  return true
end
```

FIG. 4

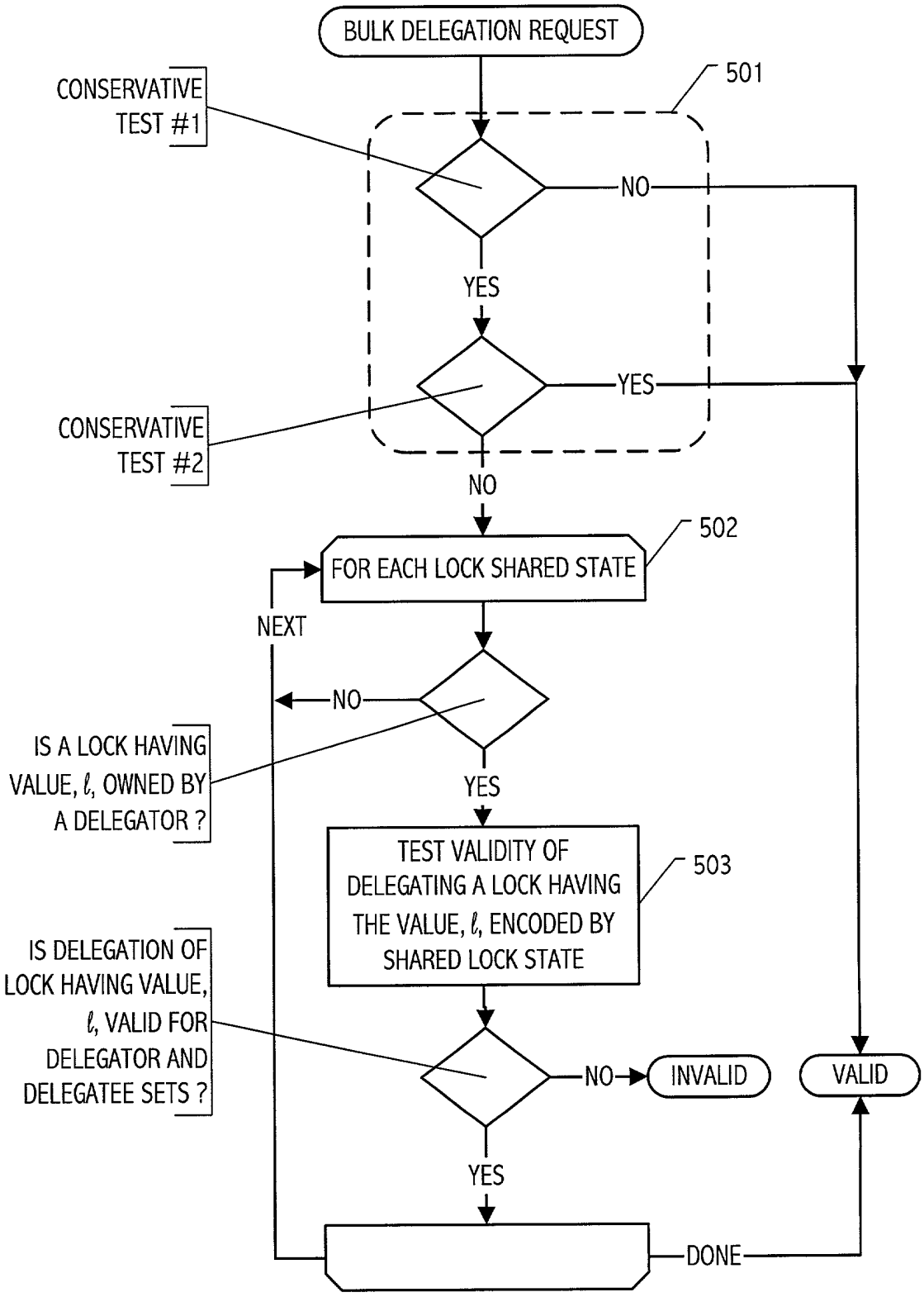
BULK DELEGATION REQUEST
501
CONSERVATIVE TEST #1
NO
YES
YES
CONSERVATIVE TEST #2
NO
502
FOR EACH LOCK SHARED STATE
NEXT
NO
IS A LOCK HAVING VALUE, ℓ, OWNED BY A DELEGATOR ?
YES
TEST VALIDITY OF DELEGATING A LOCK HAVING THE VALUE, ℓ, ENCODED BY SHARED LOCK STATE
503
IS DELEGATION OF LOCK HAVING VALUE, ℓ, VALID FOR DELEGATOR AND DELEGATEE SETS ?
NO
INVALID
VALID
YES
DONE

FIG. 5

```
// Determine the validity of a bulk lock delegation
boolean isValid(delegators, delegatees)
begin
602 ~ if wset ∩ delegators = ∅
        return true
603 ~ else if ( ∀ t_d ∈ delegatees, ∀ t_s ∈ delegators, ∀ C ∈ {rw, wr, ww},
              (ICW(t_s, C) − (delegators ∪ {t_d})) ⊆ ICW(t_d, C)
        // if at least one of the delegated lock is a write lock, the request
        // is valid only if the delegatees can ignore all conflicts which each other
        if ∃ l, Owners(l, Write) ∩ delegators ≠ ∅
          return (∀ t ∈ delegatees, ∀ c ∈ {rw, wr, ww}, delegatees ⊆ ICW(t, c))
        else
          return true
        endif
      else
        // The two conservative tests have failed
        foreach l in TSLS
          if Owners(l, W) ≠ ∅ ∧ (∃M, Owners(l, M) ∩ delegators ≠ ∅)
601 ~       if ¬isValid(delegators, delegatees, l)
              return false
            endif
          endif
        end
      endif
      return true
    end
```

FIG. 6

Atty. Dkt. No. 004-5639
1st Inventor: Laurent P. Daynes
Attorney: David W. O'Brien, (512) 347-9030